Computers, Materials & Continua
DOI:10.32604/cmc.2021.015614
Article

Tech Science Press

A Deep Learning-Based Recognition Approach for the Conversion of Multilingual Braille Images

Abdulmalik AlSalman[1], Abdu Gumaei[1,*], Amani AlSalman[2] and Suheer Al-Hadhrami[1]

[1]Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh, 11543, Saudi Arabia
[2]Department of Special Education, College of Education, King Saud University, Riyadh, 11543, Saudi Arabia
[*]Corresponding Author: Abdu Gumaei. Email: abdugumaei@gmail.com
Received: 30 November 2020; Accepted: 17 January 2021

Abstract: Braille-assistive technologies have helped blind people to write, read, learn, and communicate with sighted individuals for many years. These technologies enable blind people to engage with society and help break down communication barriers in their lives. The Optical Braille Recognition (OBR) system is one example of these technologies. It plays an important role in facilitating communication between sighted and blind people and assists sighted individuals in the reading and understanding of the documents of Braille cells. However, a clear gap exists in current OBR systems regarding asymmetric multilingual conversion of Braille documents. Few systems allow sighted people to read and understand Braille documents for self-learning applications. In this study, we propose a deep learning-based approach to convert Braille images into multilingual texts. This is achieved through a set of effective steps that start with image acquisition and preprocessing and end with a Braille multilingual mapping step. We develop a deep convolutional neural network (DCNN) model that takes its inputs from the second step of the approach for recognizing Braille cells. Several experiments are conducted on two datasets of Braille images to evaluate the performance of the DCNN model. The first dataset contains 1,404 labeled images of 27 Braille symbols representing the alphabet characters. The second dataset consists of 5,420 labeled images of 37 Braille symbols that represent alphabet characters, numbers, and punctuation. The proposed model achieved a classification accuracy of 99.28% on the test set of the first dataset and 98.99% on the test set of the second dataset. These results confirm the applicability of the DCNN model used in our proposed approach for multilingual Braille conversion in communicating with sighted people.

Keywords: Optical Braille recognition; OBR; Braille cells; blind; sighted; deep learning; deep convolutional neural network

1 Introduction

The World Health Organization (WHO) reports that at least 2.2 billion individuals worldwide are visually impaired. One billion of these cases could have been avoided or have impairments that have yet to be addressed [1]. The prevalence of visual impairments and blindness is increasing every year. The 2019 Annual American Printing House (APH) report for the blind included an estimate that 55,249 students of all ages were legally blind [2]. The visually impaired and blind cannot write and read texts. Instead, they use the Braille system. Even though many people from different countries worldwide use Braille to communicate with each other, Braille is not a language. It is a system that enables visually impaired and blind people to access and respond to any written documents [3]. Visually impaired and blind individuals can read the Braille system by fingers to trace raised dots, while sighted people can read it using their eyes.

The Braille system consists of cells with six raised dots, and each raised dot has a number from one to six organized in two columns. It is vital to allow visually impaired people to keep pace with the world around them. Providing Braille-assisted technology and integrating it in daily living are necessary to make the lives of visually impaired people more comfortable and efficient for communicating with others.

Many blind people worldwide have used the Braille system, but most sighted people do not know or cannot understand the Braille documentation of visually impaired people. Sighted people also need to quickly access the Braille system to interact and help individuals with visual impairments. In medicine, medical information is printed in the Braille system for the blind, but not all visually impaired people know the Braille system. Therefore, assistive technology would be useful to overcome this issue [4]. Assistive technology would also be helpful to convert Braille symbols into texts or voices to support students in regular schools or universities. Sighted people may also face difficulty understanding and reading Braille symbols that visually impaired people print as Braille scripts in different sciences such as math or chemistry [5].

Traditional optical character recognition (OCR) systems of natural languages cannot recognize Braille images because Braille cells do not consist of continuous strokes like the characters of natural languages [6]. The method of capturing and processing Braille documents and converting them into natural language or even editable Braille symbols is called an Optical Braille Recognition (OBR) system. In general, a typical OBR system has three main phases: image capturing, image segmentation, and Braille-to-text representation. In the past, researchers captured Braille images using a scanner that cannot handle distortions in the paper, such as defects or stains [7–10].

We propose a deep learning-based approach for a multilingual OBR system since the existing deep learning-based OBR approaches have been designed for a single language and still need some improvements to be more accurate.

The rest of this paper is organized as follows: Section 2 describes related studies in the literature and their limitations. Section 3 details the materials and methods used in this research, including a description of the datasets and the proposed approach. Section 4 presents the experiments and discusses the evaluation measures adopted to assess the recognition of Braille cells using the DCNN model. Finally, Section 5 presents the conclusions and future work of the study.

2 Literature Review

The development of OBR systems has been an active field of research for many years [4]. Existing OBR technologies that interpret the images of Braille into texts are expensive, non-portable, and outdated. These types of constraints apply to processes such as correcting Braille

image for skewness in OBR [11,12] and preservation and reproduction of Braille documents for the blind [13]. Isayed and Tahboub reported that only limited research on OBR has been performed using portable devices such as cameras and smartphones to capture Braille images [14].

Each natural language has its Braille system. The state-of-the-art approaches created a standalone model for each Braille system. Murray and Dias proposed a solution for the low quality of scanned Braille documents by making the scan focus on a small part of the Braille document [15]. This method is laborious and slow. Furthermore, scanners work on a flat and thin surface like standard papers, which are not similar to Braille papers. Therefore, researchers used portable image capturing on devices that included smartphones and cameras.

The captured image is pre-processed to separate the raised dot from the background. The standard technique used for image pre-processing is edge detection that differentiates the discontinuities in the brightness of the image [16]. However, this technique is insufficient because the images captured by a camera suffer from distortions and glare caused by camera lighting and angles. Methods that do not depend on the brightness of images are therefore required for this task, including using matrices and grids for OBR [7–9,17]. The corresponding mapping of the Braille symbol to the text character was done by noting the absence or presence of each dot in the character cells [14].

In earlier work [7,14,17], researchers used binary codes to map the Braille symbol to a text character, and subsequently converted these binary codes to decimal codes based on some equations. However, this method does not work well when the image is too noisy, especially when the noise affects the presence or absence of dots. In such cases, de-noise pre-processed methods can be used, such as converting to grayscale images or filtering the noisy images.

For instance, Kumar et al. [7] have increased the brightness of images to exceed the noise limitations. Li et al. [10] converted the RGB captured images to grayscale images to provide a suitable processing method for acquired images. Luna proposed a platform based on the ASCII code for OBR using Tesseract [18]. The image was captured using a scanner, and it was subsequently pre-processed using thresholding and filtering methods.

In the last few years, machine learning (ML) methods have been used to learn from a noisy image to improve the performance of traditional image-enhancement methods. In addition, ML methods have been used to enhance the effectiveness of OBR systems. Support vector machine (SVM), multi-layer perceptron (MLP), and probabilistic neural network (PNN) are traditional ML-based models that have been used to convert Braille images into texts by Li et al. [10], Morgavi et al. [8], and Wong et al. [19], respectively.

Recently, deep learning-based OBR models have been proposed for developing OBR systems and have achieved significant performance, including the models used by Hsu [4], Li et al. [20], Baumgärtner et al. [5], Kawabe et al. [21], and Shokat et al. [22]. In addition, Shokat et al. [23] presented a comprehensive survey analysis of OBR methods. Hsu [4] proposed a CNN model for recognizing 37 English Braille symbols. The model achieved 98.73% accuracy on a dataset with 26,724 images. Their method does not have image segmentation to partition the Braille image page into multiple lines. Li et al. [20] proposed a BraUNet segmentation framework for Braille recognition based on the encoder-decoder network. The dataset they used was limited, but they achieved a 99.66% f-score. In their study, the dataset had only 114 images divided into 74, 10, and 30 images for training, validation, and testing, respectively. A total of 74 images are not enough for deep learning, and 30 images may not give a fair evaluation of the robustness of the model. Baumgärtner et al. [5] used a dataset of German Braille symbols to propose a

mobile OBR based on a deep learning classification. They used Faster R-CNN and both CNN and long short-term memory (LSTM) for line detection and model classification, respectively. Kawabe et al. [21] proposed a model that recognizes the Japanese Braille system needed to convert old Braille books to electronic books. The model is based on the Caffe framework and AlexNet. It achieved 98%–99% accuracy for different executions. However, the method did not perform well for cell detection. Shokat et al. [22] created a digital English Braille character dataset containing 1284 images divided into 858 images and 390 images for training and validation. The authors proposed traditional and deep learning models and showed that the models based on deep learning, including sequential CNN and GoogLeNet pre-trained models, outperformed traditional methods such as Naïve Bayes (NB), SVM, Decision Tree (DT), and K Nearest Neighbors (KNN). They achieved 95.8% accuracy with this approach.

Limitations remain for the current state-of-the-art deep learning models and methods, and they cannot convert the Braille images effectively into multilingual texts or voices. These methods are not efficient for converting extensive lengths of Braille due to the use of line-by-line conversion. They also do not use image segmentation with deep learning algorithms to partition the Braille image page into multiple lines. Therefore, we propose a more effective approach that incorporates image segmentation with a deep convolutional neural network (DCNN) model to convert Braille images into their corresponding multilingual text. The resulting text is readable by sighted people or used by text-to-voice applications, to facilitate the communication between sighted and visually impaired or blind people.

3 Proposed Approach

A high-level flowchart of the proposed approach is shown in Fig. 1. It contains the separate steps of the OBR process in sequential order. These steps are Braille image acquisition and pre-processing, Braille cell cropping, Braille cell recognition, and Braille multilingual mapping. The first step is used to acquire the Braille image as an input, which is then processed using image-processing methods. The outputs of the first step are a preprocessed image with its segmented version.

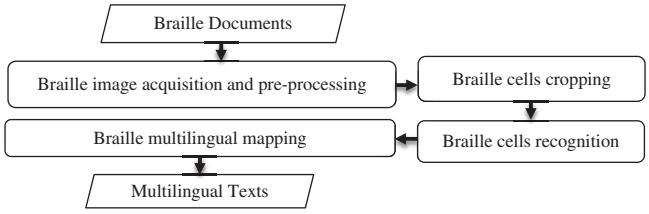


Figure 1: A high-level flowchart of the proposed approach

The segmented image in the Braille cells is cropped using the character positioning cropping algorithm (CPCA) (Braille cell cropping step). Next, in the Braille cell recognition step, a DCNN model is employed to recognize the Braille cell that resulted from the previous step. Finally, in the Braille multilingual mapping step, the labels of recognized cells, which represent the combinations of Braille dots, are used to search a lookup table to obtain the corresponding characters and symbols of the selected language.

The mapped characters and symbols are subsequently concatenated to output the sentences as a text or voice. The pseudocode for the proposed approach is shown in Algorithm 1. Lines

7–12 depict how the model classifies Braille cells and how the corresponding characters of the language are mapped and concatenated to form the natural language text.

Algorithm 1: OBR-based mapping language pseudocode

```
1. procedure Braille-To-Text Mapping Algorithm
2. OrgImgs ← ImgageAcquisition();
3. [PreImgs, SegImgs] ← ImagePreprocessing(OrgImgs);
4. (TrainSet, TrnLabels, TestSet, TstLabels) ← Split(BrailleCells)
5. DCNNmodel ← Initialize(DCNNStructure);
6. TrainedDCNNmodel ← DCNNmodel.Train (TrainSet, TrainLabels);
7. Text ←"";
8. Indicator ← "";
9. foreach BrailleCell inaTestSet do
10. TestLabel ← TrainedDCNNmodel.Classify(BrailleCell );
11. SelectedLang ← LanguageSelection();
12. if (Indicator ≠ "#") AND (Indicator ≠ CAPS)
13. OutMap ← Mapping(AlphabetPunctuTable, SelLang, TstLabel);
14. Text ← Concatenating(Text, OutMap);
15. elseif (Indicator = #)
16. OutMap ← Mapping(NumbersTable, SelLang, TstLabel);
17. Text ← Concatenating(Text, OutMap);
18. elseif (Indicator = "CAPS") AND (SelectedLang = "English")
19. OutMap ← Mapping(AlphabetPunctuTable, SelLang, TstLabel);
20. Text ← Concatenating(Text, Uppercase(OutMap));
21. end if
22. Indicator = OutMap;
23. end for
24. return Text;
25. end procedure
```

The two main steps for developing an effective OBR-based multilingual mapping task out of all the approach steps are cropping Braille cells using the CPCA and classifying them based on the DCNN model (Fig. 2).

Each Braille symbol that was cropped out using the CPCA was separately fed into the DCNN model. The output character of each symbol was then concatenated with the other mapped characters to produce the specified language text.

3.1 Braille Image Acquisition and Preprocessing

In this step, the original Braille images are captured as inputs using specific devices such as digital cameras, scanners, or cameras of mobile phones. A simple preprocessing step is applied to segment Braille dots from the image and correct the direction of the Braille cells if there is any skewing during the acquisition phase of the captured images of Braille cells. This step is performed before partitioning the captured Braille image into Braille cells in the cropping step and recognizing them using the DCNN model. We use the method proposed earlier for segmenting Braille dots [24], an approach that segments the image using the estimated thresholds values resulting from the Beta distribution. In this method, the Braille image histogram is used to model

the shape parameters of the Beta distribution and to calculate the threshold values used for image segmentation. The skewing problem is solved by rotating the original image until the parallel lines, which run horizontally and vertically through the segmented image, exactly intersect with the Braille dots [24]. The white-space around the Braille image is also cropped in the preprocessing step to achieve an optimal partitioning of Braille cells. Other preprocessing methods, including filtering, grayscale conversion, and image enhancement, are unnecessary for the DCNN model due to its ability to recognize noisy Braille cells after the model is trained on different imperfect Braille images. The output of this step shows a captured preprocessed image with its segmented version of a Braille document (Fig. 3).

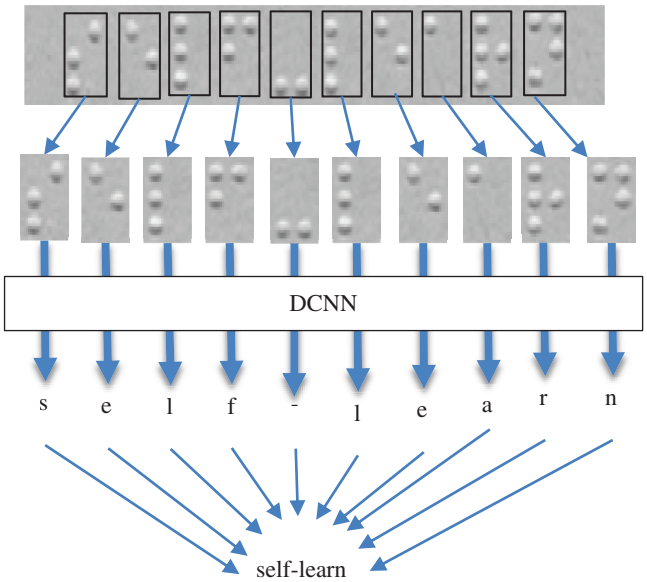
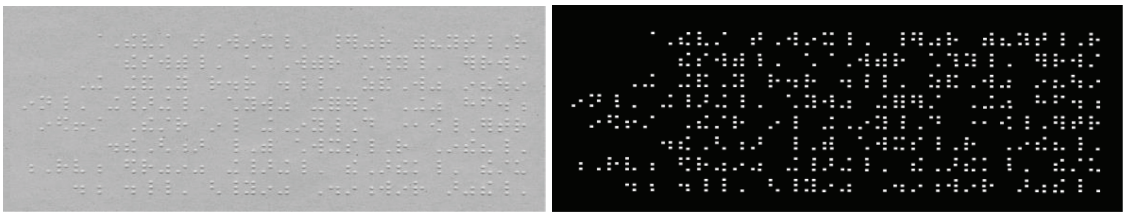


Figure 2: Construction process for the two main steps of the proposed approach



(a) (b)

Figure 3: The output of the Braille acquisition and preprocessing step: (a) original scanned Braille image and (b) its segmented image

3.2 Braille Cell Cropping

In this step, a CPCA is proposed to extract Braille cells from the captured images. This step uses the segmented image resulting from the previous step and the standard measurement of the Braille cells. The CPCA takes the first and the last positions of the Braille dots in the

segmented image that results from the previous step. These two Braille dots can be represented by $(xMin, yMin)$ and $(xMax, yMax)$. Next, the Braille cells in the processed original Braille image are cropped by using a set of tasks written as follows.

- Task 1: Determine the number of rows and columns in the defined region of the segmented image. The average distances of and between Braille cells in the Braille document can be identified as shown in Fig. 4. The number of rows and columns based on these distances is calculated as follows:

$$NumOfRows = (yMax - yMin)/r \tag{1}$$

$$NumOfCols = (xMax - xMin)/c \tag{2}$$

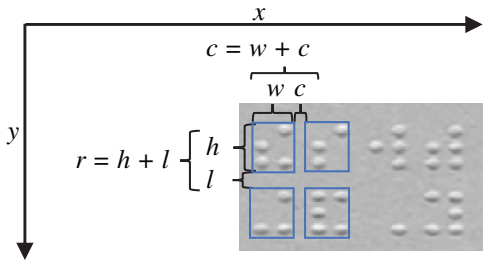


Figure 4: Distance measurements of Braille cells

- Task 2: Find the coordinates of any cell by using the row and column numbers computed as follows:

$$i = yMin + (RowNum - 1) * r \tag{3}$$

$$j = xMin + (ColNum - 1) * c \tag{4}$$

- Task 3: Finally, use the coordinates resulting from the previous task to crop the Braille cells of the original Braille image and use as inputs for the next step of the proposed approach.

3.3 Braille Cell Recognition

This step recognizes the Braille cells based on an end-to-end DCNN model due to its ability to learn image features better than traditional methods for image feature extraction [25–28]. Moreover, DCNN models have achieved high accuracy results compared with other deep learning models for several computer vision-based tasks [29]. The structure of the DCNN model is shown in Fig. 5. It consists of three blocks. The first and the second blocks are convolutional blocks. Each contains a two-dimensional convolutional layer, a two-dimensional max-pooling layer, and a leaky ReLU with a negative slope value. The third block is a dense block that consists of a fully connected linear layer followed by a leaky ReLU and a final fully connected linear layer. The convolutional and max-pooling layers are consecutively connected to extract the features of the input Braille cell image. The last two fully connected dense layers form a neural network (NN) classifier to perform the classification task. The linear layers use a linear transformation on the incoming features.

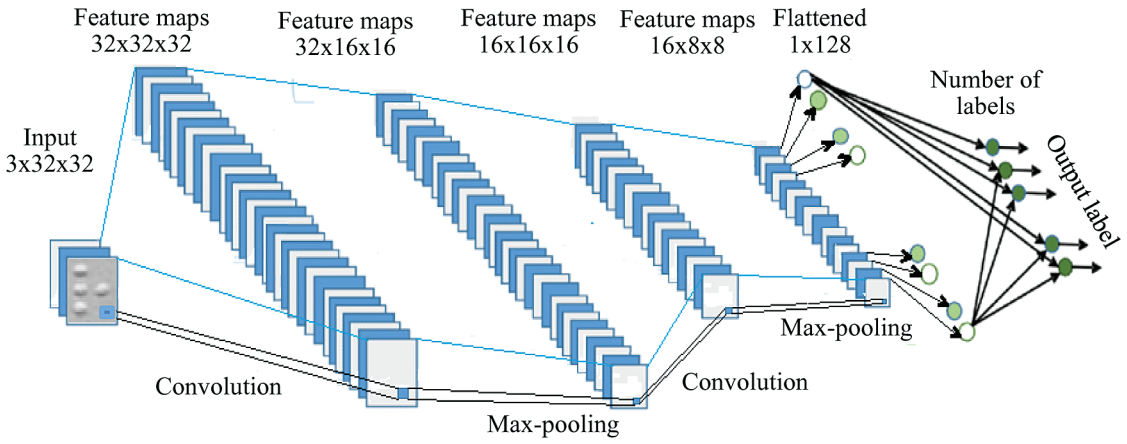


Figure 5: The structure of the deep convolutional neural network (DCNN) model

Our DCNN model differs from other models used for recognizing Braille cells in two ways. The first difference is that the input image is resized to 32×32 instead of 28×28 to retain the vital information without deformation and to reach a high performance for the detection systems [25]. The second difference is that the input channel of the first convolution layer is 32, and the input of the second convolution layer is 16. Setting the size of the input channel of the first convolution layer to be larger than the second layer helps to effectively detect low-level features such as edges and curves [30].

Convolutional layers are the essential parts of the model structure. Each neuron of these layers is locally connected to the previous layer with weight [31]. The connected weights can form a matrix called a filter or a convolution kernel. The convolution filter can extract the features of different positions of the input image to output a feature map in which the weight of each neuron is shared. The max-pooling layer is a down-sampling process that combines the low-level visual features into higher-level and abstract visual features. This layer works to aggregate the feature information obtained by the convolutional layer at different positions. A total of 80% of each dataset is used for training the DCNN model, and the remaining 20% is divided equally into a test set and a validation set. The validation set is used for hyper-parameter selection to ensure that the parameters of the model do not over-fit the training set in the training phase. In more detail, in the training phase, the hyper-parameters of the model start with initial values and are evaluated using the validation set accuracy indices. Next, these parameters are updated with new values. Finally, the suitable values are selected based on achieving acceptable accuracy on the validation set. The values of the parameters of the DCNN model are determined in the experiments. The test set is applied to measure the performance of the model.

The labels of the Braille images are digits that represent the combinations of dots in the Braille cells. These labels are converted to binary format using a one-hot encoding method to train the DCNN model. The classified labels of the DCNN model are re-converted again to digits in the testing and validation phases.

3.4 Braille Multilingual Mapping

In this step, the classified labels of Braille cells are converted to digits that represent the combinations of Braille dots. These labels are generated from the previous step according to the input cells based on the DCNN model. The classified labels are then used to retrieve the

corresponding characters of each language from a pre-defined lookup table. This lookup table contains a number of digits for all possible combinations of Braille dots starting from "no raised dots" to "six raised dots" with their corresponding characters of the specific language. A subset of the combinations of the dots and their labels with the characters of the language are shown in Tab. 1. A subset of the combination of dots as class labels with the language numbers is shown in Tab. 2.

Table 1: Subset of dot combinations as class labels with the language characters

Braille Dot Combinations	Braille Cells	Arabic Braille	English Braille	Bangladesh Braille	India (Devanagari) Braille
0		' '	' '	' '	' '
1		'ا'	'a'	'অ'	'अ'
13		'ب'	'b'	'ব'	'ब'
2345		'ت'	't'	'ত'	'त'
1246		'ث'	'?'	'থ'	'थ'
234		'ج'	'j'	'জ'	'ज'
146		'ح'	':'	'জ্ঞ'	'घ'
1256		'خ'	'x'	'খ'	'औ'
124		'د'	'd'	'দ'	'द'
2356		'ذ'	'!'	'ধ'	'ध'
1345		'ر'	'r'	'র'	'र'
1456		'ز'	'z'	'ঝ'	'ज़'
235		'س'	's'	'স'	'स'
126		'ش'	'%'	'শ'	'श'
12356		'ص'	'&'	'ষ'	'ष'
1236		'ض'	'$'	'ড'	'ड'
23456		'ط'	'('	'ট'	'ट'
123456		'ظ'	'='	'ঢ'	'ढ'
13456		'ع'	')'	'ঢ়'	
136		'غ'	'>'	'ঘ'	'घ'
123		'ف'	'f'		'फ़'
12345		'ق'	'q'	'ক্ষ'	'क्ष'
15		'ك'	'k'	'ক'	'क'
135		'ل'	'l'	'ল'	'ल'
156		'م'	'u'	'উ'	'उ'
1245		'ن'	'n'	'ন'	'न'
134		'ه'	'h'	'হ'	'ह'
2346		'و'	'w'	'ঠ'	'ठ'
23		'ي'	'i'	'ই'	'इ'
14			'e'	'এ'	'ए'
236		'؟'	ow	'ঔ'	'औ'
36		'-'	en	'য়'	'ऐ'
56		'؛'			
6			CAPS		
2456		'#'	'#'	'#'	'#'

The output of this step is a natural language text that contains the corresponding characters of classified Braille cells. The mapped texts can be stored in a file that can be read by

voice-reading software such as Natural reader or eSpeak [32]. Alternatively, the mapped texts can be stored for further processing.

Table 2: Subset of dot combinations as class labels with the language numbers

Braille Dots Combinations	Braille Cells	Arabic Braille	English Braille	Bangladesh Braille	India (Devanagari) Braille
1		1	1	1	1
13		2	2	2	2
12		3	3	3	3
124		4	4	4	4
14		5	5	5	5
123		6	6	6	6
1234		7	7	7	7
134		8	8	8	8
23		9	9	9	9
234		0	0	0	0

4 Experiments and Discussion

Two experiments were conducted on two real Braille datasets using a set of evaluation metrics to evaluate the effectiveness of the proposed approach. The following subsections describe the datasets, the evaluation metrics, and the experimental results with a discussion. We provide the experimental results of the model on two test sets selected randomly to focus in detail on the classification functionality of the model. Finally, we report the average classification result on the two datasets.

4.1 Datasets Description

The datasets of the Braille images, which are used to evaluate the DCNN model of the proposed approach, are collected from online sources, signs, and rare books, and some noise is added to the images of the dataset [33]. Next, Braille cells of these images are cropped and labeled independently (Fig. 6). We use a dataset (dataset 1) that contains 1,404 labeled images of Braille cells for 27 symbols representing the alphabet characters. Moreover, we use another dataset (dataset 2) that consists of 5,420 labeled images of 37 Braille symbols, which represent the alphabet characters, numbers, and punctuation. Therefore, the first dataset has 27 class labels: 26 for the alphabet characters and one for space (Fig. 7). The second dataset contains 37 class labels: 26 for the alphabet characters, 10 for punctuations, and one for space (Fig. 8). The first ten labels of dataset 2 can be used for labeling the numbers after recognizing the label of the hash character. The uppercase characters of the English language can also be labeled by recognizing the CAPS label of dataset 2. As stated before, 80% of each dataset is used for training, and the remaining 20% is divided equally into a test set and a validation set. Therefore, the first dataset is divided into 1,124 images in the training set, 144 images in the test set, and 144 images in the validation set. In contrast, the second dataset is split into 3,165 images in the training set, 1,860 images in the test set, and 395 in the validation set.

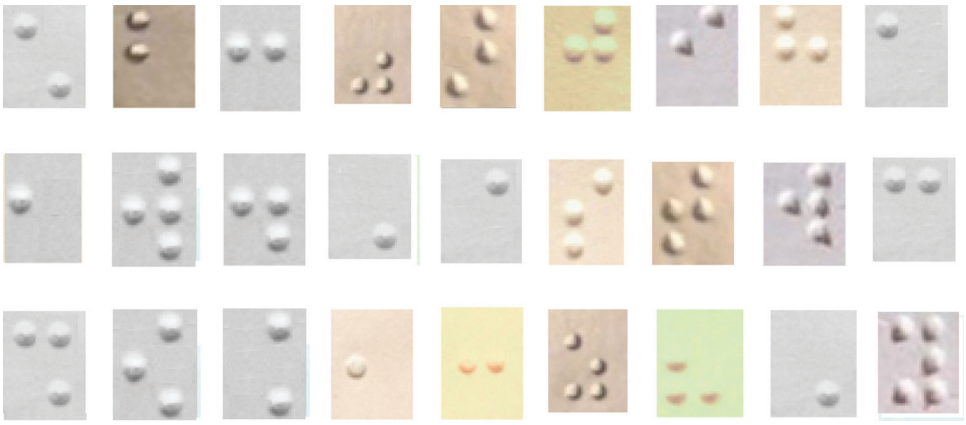


Figure 6: A set of Braille cell images from the collected datasets

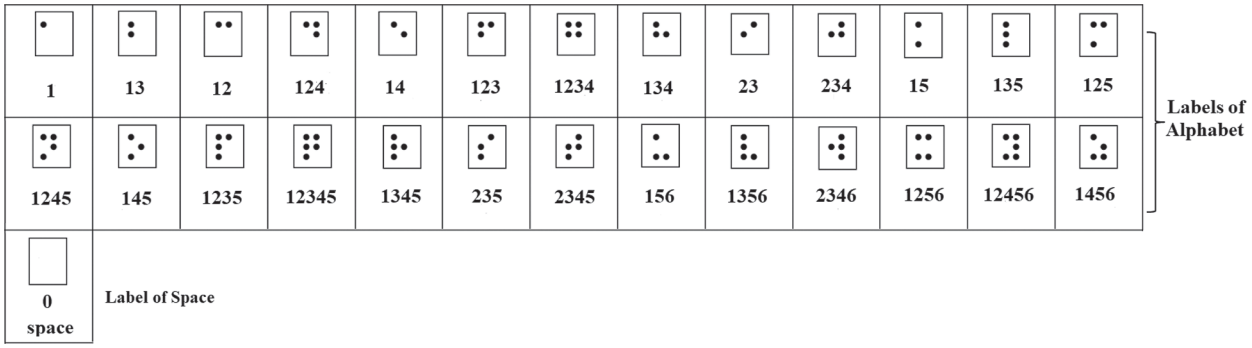


Figure 7: The 27 class labels of the first dataset

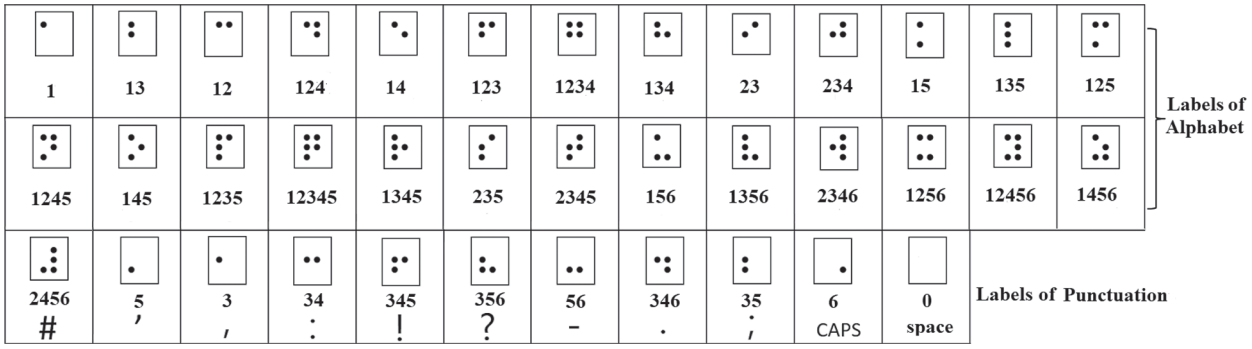


Figure 8: The 37 class labels of the first dataset

4.2 Evaluation Metrics

In the evaluation stage for the ML model, it is often necessary to use various metrics for validating the performance of the model. Accuracy, precision, recall, and F1-score are considered

better performance measures for evaluating classification problems [34]. The accuracy metric is the most primitive evaluation index in classification problems, and t can be defined as the percentage of the correct results in the total sample. The precision metric is the percentage of positive instances classified correctly from the test set; the recall metric is the percentage of positive instances that are classified as positive. F1-score is the weighted harmonic average of precision and recall. They are calculated using the following equations:

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})} \tag{5}$$

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \tag{6}$$

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})} \tag{7}$$

$$\text{F1} - \text{score} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} \ + \ \text{Recall})} \tag{8}$$

TN, TP, FN, and FP are the numbers of true negatives, true positives, false negatives, and false positives, respectively.

4.3 Results and Performance Comparison

A batch size of 144 is used for the first dataset and 64 for the second dataset after training and validating the DCNN model on both datasets. The other model parameters are initialized using 1e−3 for the learning rate, the cross-entropy method for the loss function, and the Adam optimizer for the optimization algorithm of the model. The results of the evaluation metrics on the test instances of dataset 1 are shown in Tab. 3. The results of the test instances on dataset 2 are documented in Tab. 4. Furthermore, the loss and accuracy results of the training and validation sets for both datasets during the training phase at a different number of epochs are shown in Figs. 9 and 10.

The DCNN model achieves 98.23% accuracy on the test instances of dataset 1 and 98.99% accuracy on the test instances of dataset 2 (Tabs. 3 and 4). In addition, this model yields 99.3% and 99.0% for the weighted average F1-scores on the test instances of dataset 1 and dataset 2, respectively. These results confirm the ability of the DCNN model to recognize Braille cells and validate the applicability of the approach for multilingual Braille conversion in self-learning applications.

The DCNN model achieves comparable performance on the training and validation sets for both datasets in the plot of loss results (Fig. 9). We can also see that the plots of training loss and validation loss start to depart consistently, which indicates that the training could be stopped at an earlier epoch.

The DCNN becomes a trained model when the number of epochs reaches 20 as is evident from the plot of accuracy results in Fig. 10. The accuracy of both datasets is still stable as the number of epochs increases, which shows similar progress on both datasets.

We train and test both models on the same datasets to compare the accuracy and F1-score results of our DCNN model with the model proposed in recent work [4]. The accuracy and F1-score results on the same test samples of dataset 1 and dataset 2 are shown in Tab. 5.

Table 3: Results of accuracy, precision, recall, and F1-score on the test instances of dataset 1

Class label	Precision	Recall	F1-score	Number of instances
1	1.000	1.000	1.000	2
13	1.000	1.000	1.000	3
12	1.000	1.000	1.000	2
124	1.000	1.000	1.000	5
14	1.000	1.000	1.000	6
123	1.000	1.000	1.000	5
1234	1.000	0.750	0.857	4
134	0.857	1.000	0.923	6
23	1.000	1.000	1.000	5
234	1.000	1.000	1.000	7
15	1.000	1.000	1.000	8
135	1.000	1.000	1.000	4
125	1.000	1.000	1.000	9
1245	1.000	1.000	1.000	10
145	1.000	1.000	1.000	4
1235	1.000	1.000	1.000	1
12345	1.000	1.000	1.000	7
1345	1.000	1.000	1.000	5
235	1.000	1.000	1.000	6
2345	1.000	1.000	1.000	3
156	1.000	1.000	1.000	5
1356	1.000	1.000	1.000	6
2346	1.000	1.000	1.000	4
1256	1.000	1.000	1.000	5
12456	1.000	1.000	1.000	6
1456	1.000	1.000	1.000	5
0	1.000	1.000	1.000	7
Macro average	0.995	0.991	0.992	140
Weighted average	0.994	0.993	0.993	140
Accuracy	99.29%			140

The results highlighted in boldface font in Tab. 5 show that the DCNN model improves the performance of OBR and outperforms the model proposed in recent work. Limitations remain for our work even though the F1-scores of the proposed model are 99.30% and 99.00% for recognizing Braille cells of dataset 1 and dataset 2, respectively. Our approach is not extended to cover Braille abbreviations and contractions in the multilingual mapping step. Furthermore, the approach cannot automatically detect the language of the Braille document before mapping it to an appropriate language text. Finally, the DCNN model of the proposed approach needs to be evaluated on a large-scale dataset of Braille images for different natural languages.

Table 4: Results of accuracy, precision, recall, and F1-score on the test instances of dataset 2

Class label	Precision	Recall	F1-score	Number of instances
1	1.000	0.909	0.952	11
13	1.000	1.000	1.000	10
12	1.000	1.000	1.000	18
124	1.000	1.000	1.000	20
14	1.000	1.000	1.000	23
123	1.000	0.941	0.970	17
1234	1.000	1.000	1.000	22
134	1.000	1.000	1.000	17
23	1.000	0.944	0.971	18
234	1.000	1.000	1.000	15
15	1.000	1.000	1.000	7
135	1.000	1.000	1.000	12
125	1.000	1.000	1.000	18
1245	1.000	1.000	1.000	17
145	1.000	0.950	0.974	20
1235	1.000	1.000	1.000	18
12345	0.895	1.000	0.944	17
1345	1.000	1.000	1.000	17
235	1.000	1.000	1.000	21
2345	1.000	1.000	1.000	9
156	1.000	1.000	1.000	4
1356	0.800	1.000	0.889	4
2346	1.000	1.000	1.000	4
1256	1.000	1.000	1.000	3
12456	1.000	1.000	1.000	6
1456	1.000	1.000	1.000	4
0	1.000	1.000	1.000	1
2456	1.000	1.000	1.000	7
236	1.000	1.000	1.000	6
3	0.500	1.000	0.667	1
34	1.000	1.000	1.000	3
5	1.000	1.000	1.000	6
56	1.000	1.000	1.000	3
35	1.000	1.000	1.000	4
356	1.000	1.000	1.000	3
345	1.000	1.000	1.000	2
6	1.000	1.000	1.000	7
Macro average	0.978	0.993	0.983	395
Weighted average	0.992	0.990	0.990	395
Accuracy	98.99%			395

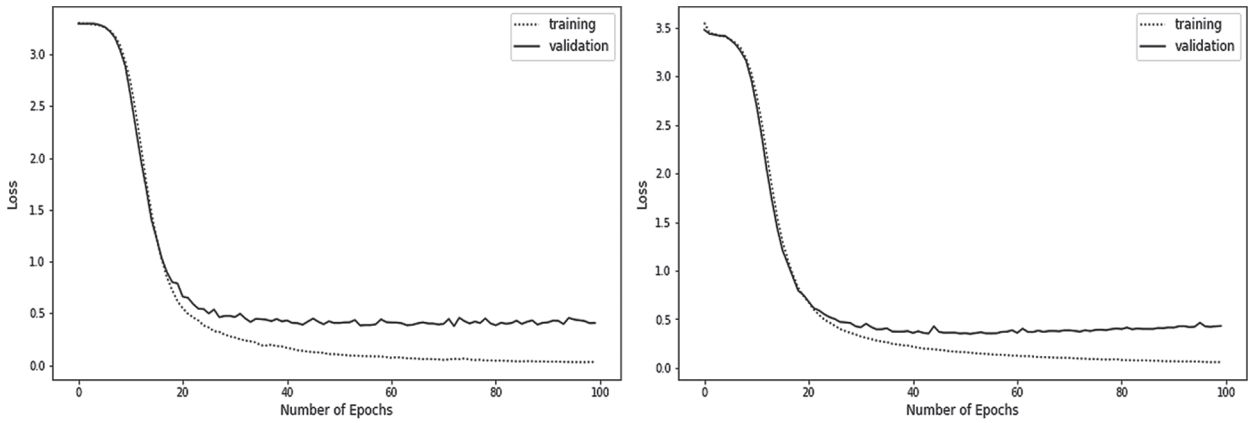


Figure 9: Training and validation loss of the DCNN model on datasets: (a) training and validation loss on dataset 1 and (b) training and validation loss on dataset 2

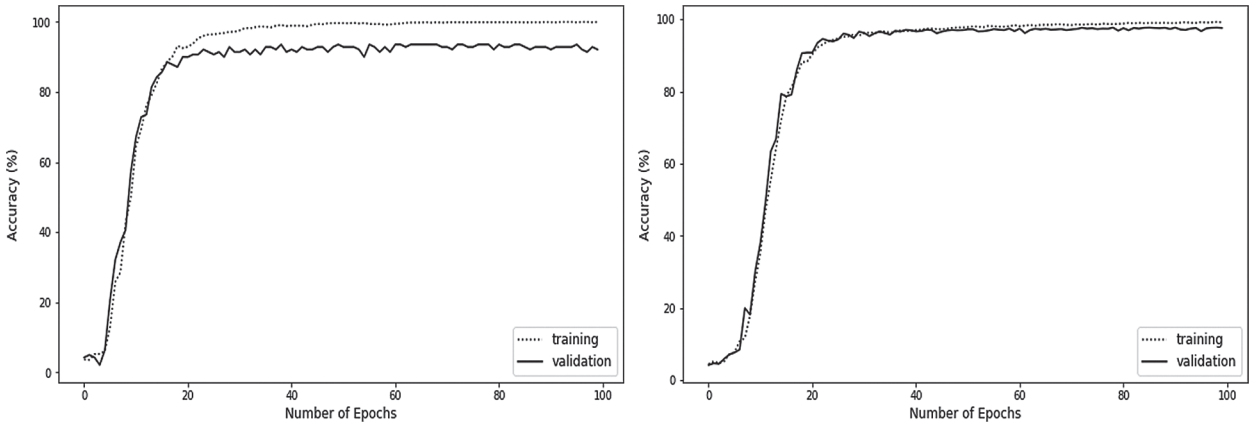


Figure 10: Training and validation accuracy of the DCNN model on datasets: (a) training and validation accuracy on dataset 1 and (b) training and validation accuracy on dataset 2

Table 5: Comparison results of accuracy and F1-score for the proposed approach and recent work on the same test samples of dataset 1 and dataset 2

Authors [Ref.], (Year)	Dataset	Accuracy (%)	F1-score (%)
Hsu [4], (2020)	Dataset 1	98.57	98.60
	Dataset 2	97.47	97.40
This work	Dataset 1	**99.29**	**99.30**
	Dataset 2	**98.99**	**99.00**

5 Conclusions and Future Work

A deep learning-based recognition approach is proposed to convert Braille images into multilingual text to facilitate communication between sighted and blind or visually impaired individuals. The approach consists of multiple steps: image acquisition and preprocessing, Braille cell cropping, Braille cell recognition, and Braille multilingual mapping. In Braille cell cropping, a CPCA is proposed to extract Braille cells from the captured images based on the standard measurement of the Braille cells. A DCNN model is developed to recognize the Braille cells. This model returns the corresponding labels for retrieving the multilingual characters from a lookup table in the next step. Several experiments are conducted on two datasets of Braille images to evaluate the performance of the DCNN model. The experimental results showed that the proposed model is able to achieve a classification accuracy of 99.28% on the test set of the first dataset and 98.99% on the test set of the second dataset. These results confirm the applicability of the proposed approach for multilingual Braille conversion in communicating with sighted people.

Currently, there is a lack of a Braille image dataset for foreign languages. In future work, we need to collect a large-scale dataset for more foreign languages and test our approach on this dataset. Moreover, we will extend the proposed approach to cover Braille abbreviations and contractions in the multilingual mapping step. We also plan to automatically detect the language by identifying the most frequently used words or sub-words in that language. For example, it is almost impossible to have a full text in English with no "the" or "a" words. Similarly, the text of the Arabic language most likely has the sub-word "ال".

Funding Statement: This project was funded by the National Plan for Science, Technology and Innovation (MAARIFAH), King Abdulaziz City for Science and Technology, Kingdom of Saudi Arabia, Award Number (5-18-03-001-0004).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] B. Swenor, V. Varadaraj, M. Lee, H. Whitson and P. Ramulu, "World health report on vision: Aging implications for global vision and eye health," *Innovation in Aging*, vol. 4, no. Suppl. 1, pp. 807–808, 2020.

[2] A. P. House, American printing house for the blind. *Annual reports, 1 October 2018–30 September*, pp. 1–28, 2019. [Online]. Available: https://www.aph.org/annual-reports/ File: Annual-Report-FY2019-accessible.pdf.

[3] S. Gayathri, B. Sujathakumari, N. S. Murthy, M. Balabhadra, N. S. Joshi *et al.,* "Digitized Braille cell: A novel solution for visually impaired," *International Journal of Information Technology*, 2020. https://doi.org/10.1007/s41870-020-00506-9.

[4] B.-M. Hsu, "Braille recognition for reducing asymmetric communication between the blind and non-blind," *Symmetry*, vol. 12, no. 7, pp. 1069, 2020.

[5] C. Baumgärtner, T. Schwarz and R. Stiefelhagen, "Image-based recognition of Braille using neural networks on mobile devices," in *Int. Conf. on Computers Helping People with Special Needs*, Springer, Lecco, Italy, pp. 346–353, 2020.

[6] S. Srinath and C. R. Kumar, "An insight into optical Braille character recognition since its conceptualisation," *International Journal of Computer Applications*, vol. 33, no. 6, pp. 1–4, 2011.

[7] C. R. Kumar and S. Srinath, "A novel and efficient algorithm to recognize any universally accepted Braille characters: A case with kannada language," in *2014 Fifth Int. Conf. on Signal and Image Processing*, IEEE, Bangalore, India, pp. 292–296, 2014.

[8] G. Morgavi and M. Morando, "A neural network hybrid model for an optical Braille recognitor," in *Int. Conf. on Signal, Speech and Image Processing 2002*, Koukounaries, Skiathos Island, Greece, 2002.
[9] S. D. Al-Shamma and S. Fathi, "Arabic Braille recognition and transcription into text and voice," in *2010 5th Cairo Int. Biomedical Engineering Conf.*, IEEE, Cairo, Egypt, pp. 227–231, 2010.
[10] J. Li and X. Yan, "Optical Braille character recognition with support-vector machine classifier," in *2010 Int. Conf. on Computer Application and System Modeling*, IEEE, Taiyuan, China, vol. 12, pp. 219–222, 2010.
[11] A. M. S. Al-Salman, A. El-Zaart and A. Gomai, "A new approach for adjusting Braille image skewness in optical Braille recognition," in *Int. Conf. on Software Engineering and Computer Systems*, Springer, Kuantan, Pahang, Malaysia, pp. 735–746, 2011.
[12] T. J. Bani-Ata and R. I. Zaghloul, "Skew correction in Braille recognition systems," *Business Management*, vol. 10, no. 3, pp. 191–196, 2018.
[13] L. J. Sisco, "Braille preservation: Recognising and respecting archival materials produced by and for the blind," *Archives Manuscripts*, vol. 43, no. 1, pp. 18–28, 2015.
[14] S. Isayed and R. Tahboub, "A review of optical Braille recognition," in *2015 2nd World Symp. on Web Applications and Networking*, IEEE, Sousse, Tunisia, pp. 1–6, 2015.
[15] I. Murray and T. Dias, "A portable device for optically recognizing Braille-part II: Software development," in *The Seventh Australian and New Zealand Intelligent Information Systems Conf.*, IEEE, Perth, Western Australia, pp. 141–145, 2001.
[16] C. Ng, V. Ng and Y. Lau, "Regular feature extraction for recognition of Braille," in *Pro. Third Int. Conf. on Computational Intelligence and Multimedia Applications. (Cat. No. PR00300)*, IEEE, New Delhi, India, pp. 302–306, 1999.
[17] S. Zhang and K. Yoshino, "A Braille recognition system by the mobile phone with embedded camera," in *Second Int. Conf. on Innovative Computing, Information and Control*, IEEE, Kumamoto, Japan, pp. 223, 2007.
[18] P. Chakraborty and A. Mallik, "An open source tesseract based tool for extracting text from images with application in Braille translation for the visually impaired," *International Journal of Computer Applications*, vol. 68, no. 16, pp. 26–32, 2013.
[19] L. Wong, W. Abdulla and S. Hussmann, "A software algorithm prototype for optical recognition of embossed Braille," *Proc. of the 17th Int. Conf. on Pattern Recognition, 2004*, vol. 2, pp. 586–589, 2004.
[20] R. Li, H. Liu, X. Wang, J. Xu and Y. Qian, "Optical Braille recognition based on semantic segmentation network with auxiliary learning strategy," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition Workshops*, Seattle, WA, USA, pp. 554–555, 2020.
[21] H. Kawabe, Y. Shimomura, H. Nambo and S. Seto, "Application of deep learning to classification of Braille dot for restoration of old Braille books," in *Int. Conf. on Management Science and Engineering Management*, Springer, Melbourne, Australia, pp. 913–926, 2018.
[22] S. Shokat, R. Riaz, S. S. Rizvi, A. M. Abbasi, A. A. Abbasi *et al.,* "Deep learning scheme for character prediction with position-free touch screen-based Braille input method," *Human-centric Computing Information Sciences*, vol. 10, no. 1, pp. 1–24, 2020.
[23] S. Shokat, R. Riaz, S. S. Rizvi, K. Khan, F. Riaz *et al.,* "Analysis and evaluation of Braille to text conversion methods," *Mobile Information Systems*, vol. 2020, pp. 1–14, 2020.
[24] A. M. S. Al-Salman, A. El-Zaart, Y. Al-Suhaibani, K. Al-Hokail and A. Gumaei, "Designing Braille copier based on image processing techniques," *International Journal of Soft Computing and Engineering*, vol. 4, no. 5, pp. 62–69, 2014.
[25] B. Kien, D. Iba, Y. Tsutsui, A. Kajihata and Y. Lei *et al.*, "Effect of image size on performance of a plastic gear crack detection system based convolutional neural networks: An experimental study," *Health Monitoring of Structural and Biological Systems IX, International Society for Optics and Photonics*, vol. 11381, pp. 113812I, 2020.
[26] S. Velliangira and J. Premalata, "A novel forgery detection in image frames of the videos using enhanced convolutional neural network in face images," *Computer Modeling in Engineering Sciences*, vol. 125, no. 2, pp. 625–645, 2020.

[27] C. L. Wu, Q. Zhang, J. Zhou, H. Yang and Y. Li, "Text detection and recognition for natural scene images using deep convolutional neural networks," *Computers, Materials & Continua*, vol. 61, no. 1, pp. 289–300, 2019.

[28] J. Chen, Z. Zhou, Z. Pan and C. N. Yang, "Instance retrieval using region of interest based CNN features," *Journal of New Media*, vol. 1, no. 2, pp. 87, 2019.

[29] X. Peng, X. Zhang, Y. Li and B. Liu, "Research on image feature extraction and retrieval algorithms based on convolutional neural network," *Journal of Visual Communication Image Representation*, vol. 69, no. 6, pp. 102705, 2020.

[30] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy *et al.,* "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, no. 11, pp. 354–377, 2018.

[31] Q. Cui, S. McIntosh and H. Sun, "Identifying materials of photographic images and photorealistic computer generated graphics based on deep CNNs," *Computers, Materials & Continua*, vol. 55, no. 2, pp. 229–241, 2018.

[32] S. Padmavathi, K. Manojna, S. S. Reddy and D. Meenakshy, "Conversion of Braille to text in english, hindi and tamil languages," *International Journal of Computer Science, Engineering and Applications*, vol. 3, no. 3, pp. 19–32, 2013.

[33] H. Gezahegn, T. Y. Su, W. C. Su and M. Ito, "An optical Braille recognition system for enhancing Braille literacy and communication between the blind and non-blind," *Review of Undergraduate Computer Science*, vol. 2018–2019, no. 2, pp. 1–5, 2019.

[34] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, pp. 6, 2020.